H. Method of Troubleshooting/Debugging One Embodiment of a System

One embodiment of the system **10** also has additional functionality that may allow a worker **155** to be deployed on a local computer **100** without accessing the compute backbone **300** infrastructure or the network **200**. To allow an applications developer **30** to debug its worker modules **195-1** to **195**-N locally on its local computer **100** (which, in one embodiment, is the development host for the applications developer **30**), the compute backbone **300** is capable of (i) providing a simplified replica of itself, including an API **190**, and (ii) initializing worker modules **195-1** to **195**-N in the same process space in which the calling application **180** resides. Such a capability may enable an applications developer **30** to debug functionality, such as persistence and parameter passing, in an environment where the developer **30** has access to all necessary information about both the calling application **180** and the environment on which it is running (i.e., the replicated functionality of the compute backbone **300**). For example, if a worker module **195** performs properly on the local computer **100**, it will also perform properly when deployed on the compute backbone **300**.

FIG. **11** illustrates certain operations performed in one embodiment of a method of running a calling application **180** in local mode. For any particular calling application **180**, an applications developer **30** may create both a worker module **195** and one or more jobs **182** (step **1910**). At initialization, the developer **30** links the calling application **180** to the API **190** file associated with local mode operation (as opposed to the API **190** file associated with network mode operation) (step **1920**). The API **190** then loads the worker module **195** into the process space of the local computer **100** (step **1930**). The API **190** ensures that a replica of all major functions performed by the compute backbone **300** (e.g., scheduling, caching, etc.) are loaded into the data storage devices **110-1** to **110**-N of the local computer **100** (step **1940**). The worker **155** is then processed on the CPU **120** of the local computer **100** (step **1950**). Unlike the parallel computing operation of network mode on the actual compute backbone **300** infrastructure, processing in local mode is accomplished sequentially, or perhaps concurrently if multithreading is used.

Although illustrative embodiments and example methods have been shown and described herein in detail, it should be noted and will be appreciated by those skilled in the art that there may be numerous variations and other embodiments which may be equivalent to those explicitly shown and described. For example, the scope of the present invention is not necessarily limited in all cases to execution of the aforementioned steps in the order discussed. Unless otherwise specifically stated, the terms and expressions have been used herein as terms and expressions of description, not of limitation. Accordingly, the invention is not limited by the specific illustrated and described embodiments and examples (or the terms or expressions used to describe them) but only by the scope of appended claims.

We claim:

1. A method, comprising:

receiving, for computation by a node computing device of a distributed computing system, a parent job configured to produce one or more descendant jobs, wherein said node computing device is one of a plurality of node computing devices of said distributed computing system;

scheduling computation of said parent job on said node computing device;

selectively rescheduling computation of a job other than said parent job from any one of said plurality of node computing devices to another of said node computing devices; and

preventing rescheduling of said parent job unless each of said descendant jobs is completed or terminated.

2. The method of claim **1**, said distributed computing system further comprising a persistent data storage queue in communication with said node computing device, wherein a minimum availability of said distributed computing system is defined by an availability of said persistent data storage; and wherein said method further comprises:

storing a descendant output from each of said descendant jobs in said persistent queue for retrieval by said node computing device processing said parent job; and

accessing said persistent queue to retrieve said descendant output for use in computation of said parent job.

3. The method of claim **1**, wherein none of said node computing devices is available for computation at a time when said parent job is scheduled for computation.

4. The method of claim **1**, further comprising sending for computation each descendant job to a node computing device other than said node computing device processing said parent job.

5. The method of claim **1**, wherein said parent job comprises meta-information comprising an instruction to divide one or more of said descendant jobs from said parent job for scheduling by a scheduler server and processing by at least one of said node computing devices.

6. The method of claim **1**, further comprising terminating each of said descendant jobs upon termination of said parent job.

7. The method of claim **1**, wherein each of said node computing devices provides to a scheduler server an availability status.

8. The method of claim **1**, further comprising receiving said parent job from an application running on a local computing device.

9. The method of claim **8**, further comprising providing an output of said parent job for retrieval by said application.

10. The method of claim **1**, further comprising storing a descendant output from at least one of said descendant jobs in a cache for use by another of said descendant jobs or said parent job.

11. The method of claim **1**, wherein said descendant job comprises meta-information comprising an identification of a compute function to be used to perform a computation for said descendant job.

* * * * *